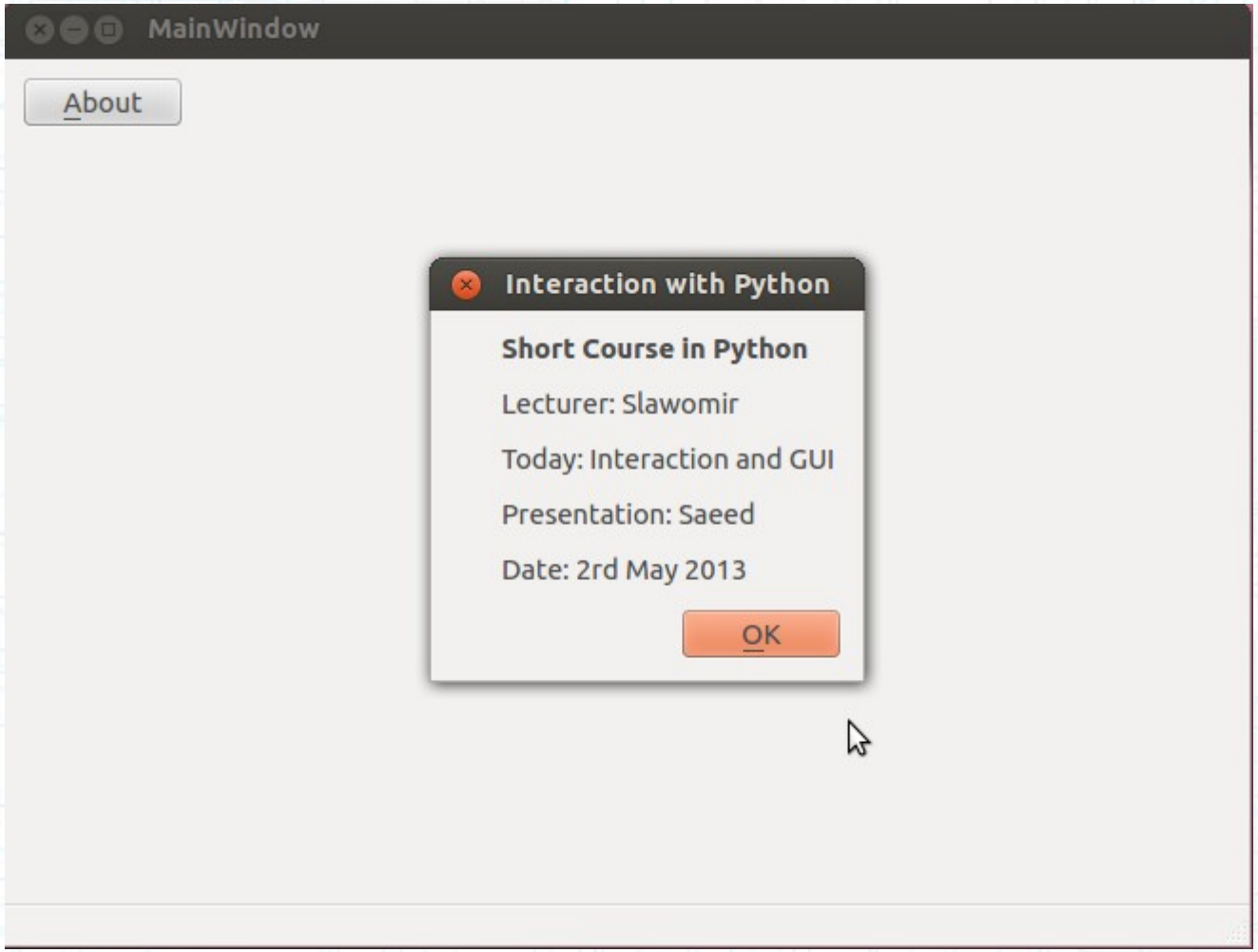
Interaction with Python

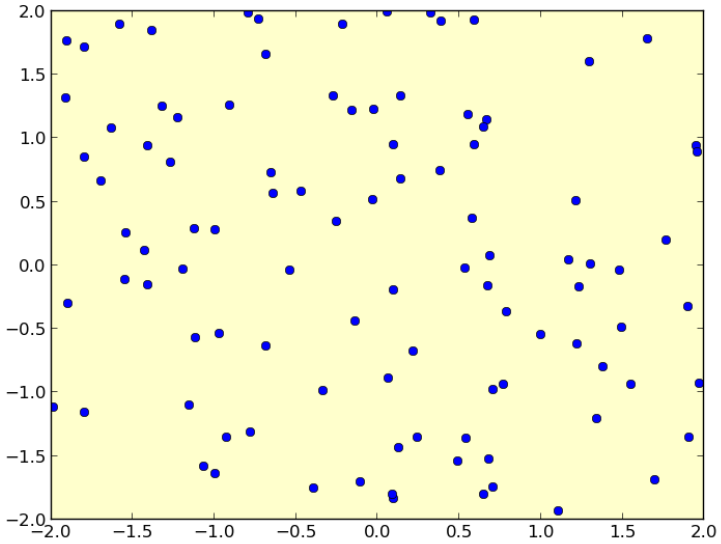


Simple widget? Try “matplotlib”

- Matplotlib.widgets
  - AxesWidget
  - Button
  - RadioButtons
  - CheckButtons
  - Slider
  - Cursor
  - HorizontalSpanSelector
  - Lasso
  - LassoSelector
  - ...

Matplotlib.widget example#1

- Cursor

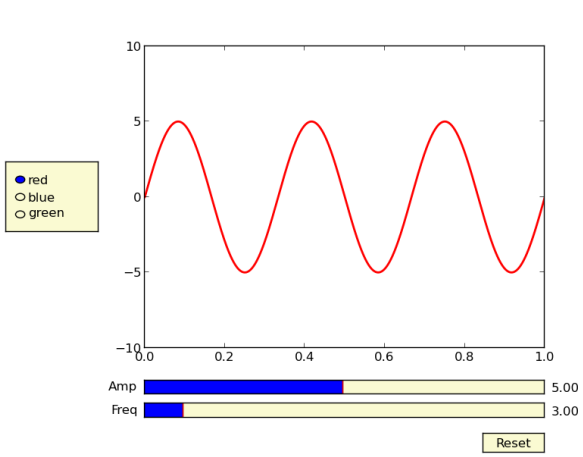


Simple Interaction

```
from matplotlib.widgets import Cursor
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, axisbg='#FFFFCC')

x, y = 4*(np.random.rand(2, 100)-.5)
ax.plot(x, y, 'o')
ax.set_xlim(-2, 2)
ax.set_ylim(-2, 2)

cursor = Cursor(ax, useblit=True, color='red', linewidth=2 )

plt.show()
```

Matplotlib.widget example#2

- Slider example:




```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button, RadioButtons
```

```
ax = plt.subplot(111)
plt.subplots_adjust(left=0.25, bottom=0.25)
t = np.arange(0.0, 1.0, 0.001)
a0 = 5
f0 = 3
s = a0*np.sin(2*np.pi*f0*t)
l, = plt.plot(t,s, lw=2, color='red')
plt.axis([0, 1, -10, 10])
```

Initialization & Plotting of a "sin" wave

```
axcolor = 'lightgoldenrodyellow'
axfreq = plt.axes([0.25, 0.1, 0.65, 0.03], axisbg=axcolor)
axamp  = plt.axes([0.25, 0.15, 0.65, 0.03], axisbg=axcolor)

sfreq = Slider(axfreq, 'Freq', 0.1, 30.0, valinit=f0)
samp = Slider(axamp, 'Amp', 0.1, 10.0, valinit=a0)
```

Sliders

```
def update(val):
    amp = samp.val
    freq = sfreq.val
    l.set_ydata(amp*np.sin(2*np.pi*freq*t))
    plt.draw()
sfreq.on_changed(update)
samp.on_changed(update)
```

Updating values on slider change

```
resetax = plt.axes([0.8, 0.025, 0.1, 0.04])
button = Button(resetax, 'Reset', color=axcolor, hovercolor='0.975')
def reset(event):
    sfreq.reset()
    samp.reset()
button.on_clicked(reset)
```

Reset button

```
rax = plt.axes([0.025, 0.5, 0.15, 0.15], axisbg=axcolor)
radio = RadioButtons(rax, ('red', 'blue', 'green'), active=0)
def colorfunc(label):
    l.set_color(label)
    plt.draw()
radio.on_clicked(colorfunc)
```

Color buttons

```
plt.show()
```

Real GUI?

- Officially presented in python.org:
  - PyGtk - Bindings for the cross-platform Gtk toolkit.
  - PyQt - Bindings for the cross-platform Qt framework.
  - TkInter - The traditional Python user interface toolkit.
  - WxPython - wxWidgets bindings for Python supporting PythonCard, Wax and other frameworks.
  - PyjamasDesktop - Bindings and a framework for the cross-platform webkit.
- “GUI Programming is, in many cases, a matter of taste. See a more extensive list on the GuiProgramming page.”
- My taste? (just intuitive though)
  - Framework: Qt
  - Binding: Pyside (LGPL-licensed)

Qt

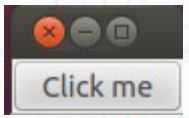
- A cross-platform application framework:
  - a widget toolkit for GUI.
  - also used for developing non-GUI programs:
    - SQL database access
    - XML parsing
    - Thread management
    - Network support
- Uses standard C++.

It's a HUGE cake

And for the moment we just need a bite

Qt Example - clickable button

```
import sys
from PySide.QtCore import *
from PySide.QtGui import *

def sayHello():
    print "Hello World!"

# Create the Qt Application
app = QApplication(sys.argv)
# Create a button, connect it and show it
button = QPushButton("Click me")
button.clicked.connect(sayHello)
button.show()
# Run the main Qt loop
app.exec_()
```

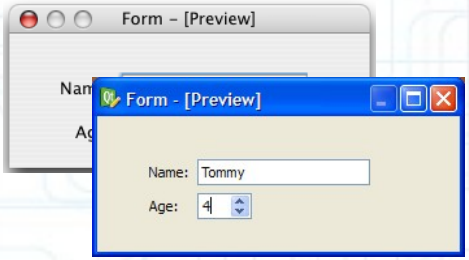


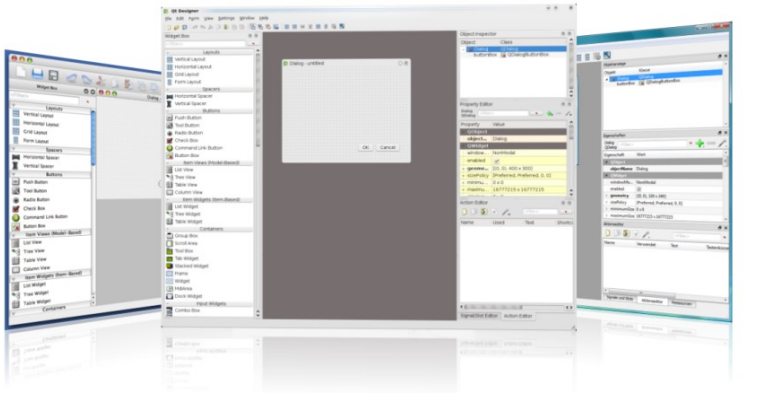
Qt Example - dialoge

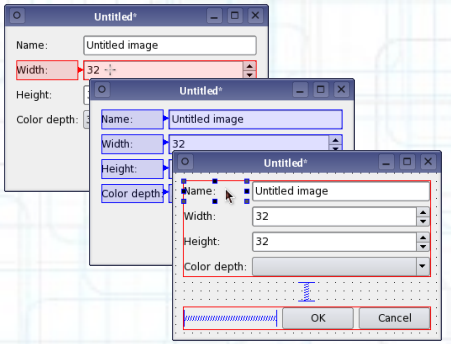
```
import sys
from PySide.QtCore import *
from PySide.QtGui import *

class Form(QDialog):

    def __init__(self, parent=None):
        super(Form, self).__init__(parent)
        # Create widgets
        self.edit = QLineEdit("Write my name here")
        self.button = QPushButton("Show Greetings")
        # Create layout and add widgets
        layout = QVBoxLayout()
        layout.addWidget(self.edit)
        layout.addWidget(self.button)
        # Set dialog layout
        self.setLayout(layout)
        # Add button signal to greetings slot
        self.button.clicked.connect(self.greetings)

    # Greets the user
    def greetings(self):
        print ("Hello %s" % self.edit.text())


if __name__ == '__main__':
    # Create the Qt Application
    app = QApplication(sys.argv)
    # Create and show the form
    form = Form()
    form.show()
    # Run the main Qt loop
    sys.exit(app.exec_())
```

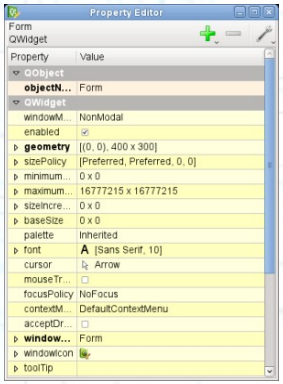
Qt Designer (Matlab's guide if you wish)

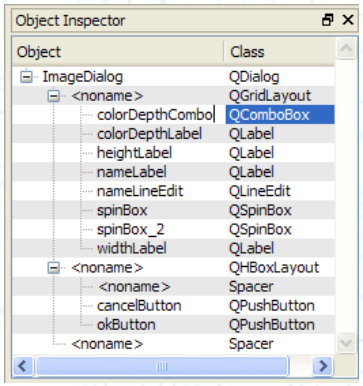
- Designer Manual

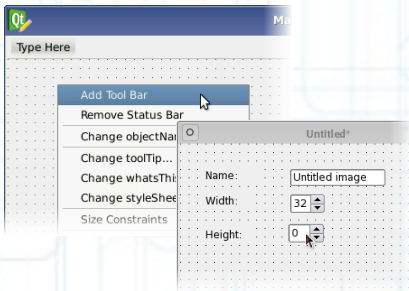












Qt Designer - design process


```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
…
..
.
```

GUIxmlfile.ui

setuptools

[link]

pyside-uic GUIxmlfile.ui -o GUIpyfile.py

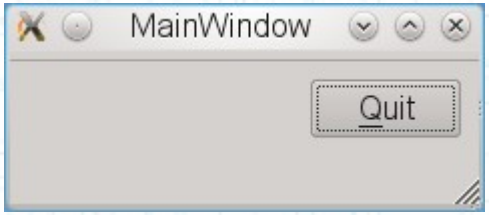
```
from PySide import QtCore, QtGui

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
```

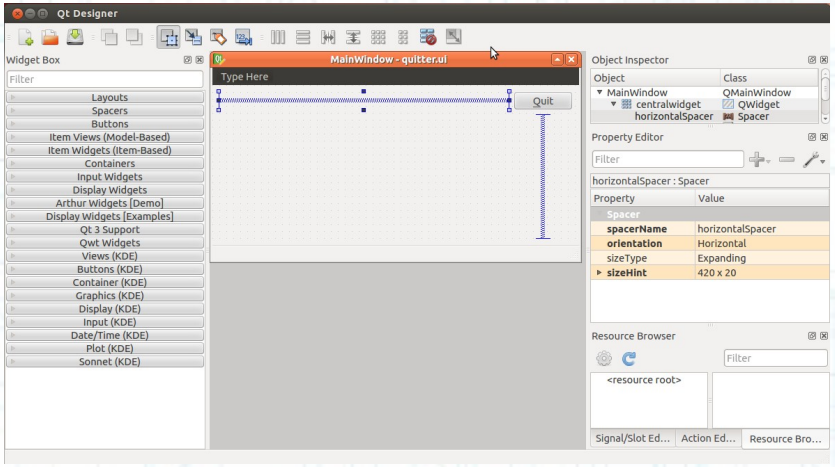
GUIpyfile.py

```
import sys

from PySide.QtGui import QMainWindow, QPushButton, QApplication
from GUIpyfile import Ui_MainWindow

class MainWindow(QMainWindow, Ui_MainWindow):
…
...
```

Mainpyfile.py

Qt Designer Example#1 - Close






```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow"
name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>526</width>
    <height>277</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget"
name="centralwidget">
   <layout class="QGridLayout"
name="gridLayout">
    <item row="0" column="0">
     <spacer name="horizontalSpacer">
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>420</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
    <item row="0" column="1">
     <widget class="QPushButton"
name="quitButton">
      <property name="text">
       <string>&amp;amp;Quit</string>
      </property>
     </widget>
    </item>    <item row="1" column="1">
     <spacer name="verticalSpacer">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>20</width>
        <height>165</height>
       </size>
      </property>
     </spacer>
    </item>
   </layout>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>526</width>
     <height>29</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections>
  <connection>
   <sender>quitButton</sender>
   <signal>clicked()</signal>
   <receiver>MainWindow</receiver>
   <slot>close()</slot>
   <hints>
    <hint type="sourcelabel">
     <x>479</x>
     <y>52</y>
    </hint>
    <hint type="destinationlabel">
     <x>262</x>
     <y>138</y>
    </hint>
   </hints>
  </connection>
 </connections>
</ui>
```

xml file

Qt Designer Example#1 - Close

```
from PySide import QtCore, QtGui


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(526, 277)
        self.centralwidget = QtGui.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtGui.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        spacerItem = QtGui.QSpacerItem(420, 20, QtGui.QSizePolicy.Expanding, QtGui.QSizePolicy.Minimum)
        self.gridLayout.addItem(spacerItem, 0, 0, 1, 1)
        self.quitButton = QtGui.QPushButton(self.centralwidget)
        self.quitButton.setObjectName("quitButton")
        self.gridLayout.addWidget(self.quitButton, 0, 1, 1, 1)
        spacerItem1 = QtGui.QSpacerItem(20, 165, QtGui.QSizePolicy.Minimum, QtGui.QSizePolicy.Expanding)
        self.gridLayout.addItem(spacerItem1, 1, 1, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtGui.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 526, 29))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtGui.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)


        self.retranslateUi(MainWindow)
        QtCore.QObject.connect(self.quitButton, QtCore.SIGNAL("clicked()"), MainWindow.close)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)


    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QtGui.QApplication.translate("MainWindow", "MainWindow", None, QtGui.QApplication.UnicodeUTF8))
        self.quitButton.setText(QtGui.QApplication.translate("MainWindow", "&Quit", None, QtGui.QApplication.UnicodeUTF8))
```

Python file from xml file

Qt Example#1 - Close

```
import sys

from PySide.QtGui import QMainWindow, QPushButton, QApplication

from ui_quitter import Ui_MainWindow

class MainWindow(QMainWindow, Ui_MainWindow):
    def __init__(self, parent=None):
        super(MainWindow, self).__init__(parent)
        self.setupUi(self)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    frame = MainWindow()
    frame.show()
    app.exec_()
```

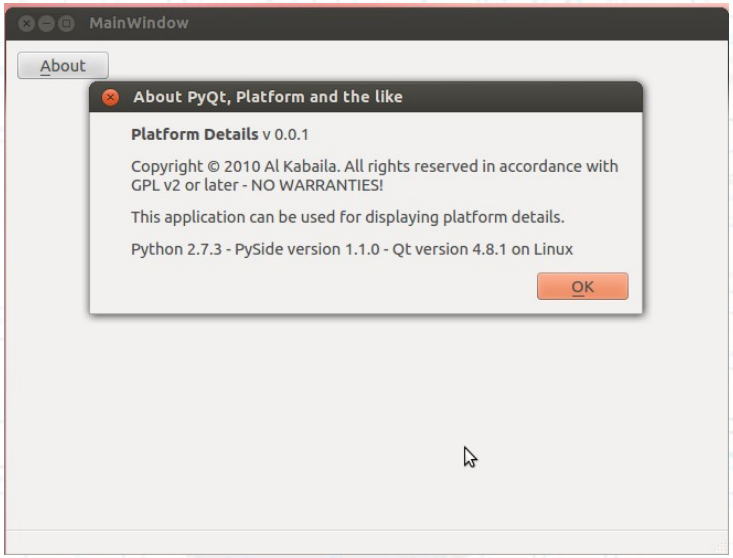
class
def
__name__

Where is close button?
It was declared in GUI

```
self.retranslateUi(MainWindow)
        QtCore.QObject.connect(self.quitButton, QtCore.SIGNAL("clicked()"), MainWindow.close)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

Qt Example#2 – About button

```
import sys
import platform

import PySide
from PySide.QtGui import QApplication, QMainWindow, QTextEdit, QPushButton,  QmessageBox

from ui_about import Ui_MainWindow

__version__ = '0.0.1'

class MainWindow(QMainWindow, Ui_MainWindow):
    def __init__(self, parent=None):
        super(MainWindow, self).__init__(parent)
        self.setupUi(self)
        self.aboutButton.clicked.connect(self.about)

    def about(self):
        '''Popup a box with about message.'''
        QMessageBox.about(self, "About PyQt, Platform and the like",
                """<b>Platform Details</b> v %s
                <p>Copyright &copy; 2010 Al Kabaila.
                All rights reserved in accordance with
                GPL v2 or later - NO WARRANTIES!
                <p>This application can be used for
                displaying platform details.
                <p>Python %s -  PySide version %s - Qt version %s on %s""" % (__version__,
                platform.python_version(), PySide.__version__,  PySide.QtCore.__version__,
                platform.system()))

if __name__ == '__main__':
    app = QApplication(sys.argv)
    frame = MainWindow()
    frame.show()
    app.exec_()
```



References and useful links

- Qt Designer, how to use? It requires some time. Follow these:
  - Creating a Qt Widget Based Application
  - How to use Qt Creator to design Graphical Interfaces for PySide
- Extra:
  - Video Tutorials
  - PySide Documentation
    - PySide Tutorials
  - More Examples
  - PySide Pitfalls

Ideas for practicing

- GUI#1
  - A button to load a sqlite file.
  - Plotting the data inside the database.
  - Optional: possibility of tweaking data from database to plot, like slider to change the amplitude of the signal.
  - Optional: listing all available table inside database and providing a choosing option.
- GUI#2
  - A button to load an image.
  - Showing image in one “axes”.
  - Plotting histogram of the image in another “axes”. (with continues line and patching.)
  - Optional: Possibility to choose between color and BW histogram.

The end

Thank you.